Teaching Statement

Throughout my academic and industry experiences I embraced any opportunity to learn the abstract ideas, fundamental theories, and key methodologies of new areas. I was enthralled with lectures and textbooks that concisely curated decades of research, enabling me to stand on the shoulders of giants that came before. As a researcher, I now contribute to the new ideas, theories, and methodologies. As a teacher, I have the honor of giving back, distilling knowledge for and instilling curiosity into the next generation of students.

Teaching Interests: My dissertation research studies parallel computer architecture, parallel programming models, and compiler-aided parallelization. For nearly two years I was a software engineer at a startup producing a distributed file-sharing system. Therefore, at the undergraduate level, I am most qualified to teach courses in computer architecture and organization, programming, performance-oriented programming, parallel programming, optimizing compilers, and systems software. At the graduate level, I can offer advanced courses in computer architecture and parallel systems.

Teaching Experience: I have several years of teaching experience, from courses in undergraduate- and graduate-level computer science and engineering, to music camp instruction and big-band jazz conducting.

I have taught in technical courses at the University of Toronto, Insight Data Science, and the Massachusetts Institute of Technology. In 2008, my final undergraduate year at U of T, I jumped at the opportunity to serve as a teaching assistant (TA) for a new course, MAT190 “Vector and Matrix Algebra”. This class provided a solid foundation of engineering mathematics for first-year students of differing mathematical backgrounds, before they undertook linear algebra the following semester. I led a weekly recitation with particular attention to active learning through problem solving. I loved the experience. The following year I was a TA for ESC103 “Engineering Mathematics and Computation”, an evolution of MAT190 that added a MATLAB laboratory to complement the presented theory. In addition to problem-solving recitations, I assisted in lab sessions by helping students to understand MATLAB programming and to develop tools to debug their programs. Through my graduate research, I gained experience in software systems and subsequently TA’d courses on operating systems (ECE353), and performance-oriented programming (ECE454). Following my time at U of T, while working at a California startup in 2012, I was invited by the Insight Data Science founder to be a mentor for the program. My students were postdoctoral fellows with computational science backgrounds. I coached them on software engineering interview coding strategies and communication. At MIT I was a TA for 6.823 “Computer System Architecture”. This is a graduate-level course that includes ISAs, out-of-order cores, virtual memory, and multicore processors. I led weekly recitations and office hours for 30 students, led review sessions before examinations, administered and graded lab assignments, and co-authored and graded exams. In Spring 2018, I gave a guest lecture on how my research can accelerate graph algorithms in 6.886 “Graph Analytics”.

Music was my introduction to teaching. From 2003 to 2009, I was an assistant percussion instructor at the National Music Camp of Canada. I taught percussion classes which culminated in a student ensemble performance for the rest of the camp. From 2008 to 2011, I was the conductor of several Engineering Jazz Ensembles at U of T. I prepared and conducted weekly rehearsals, communicating musical ideas to over 20 lively engineering students, leading to numerous acclaimed performances each year. My leadership in music at U of T Engineering earned me the L.E. Jones Award of Distinction, presented to a graduating engineering student who made a significant contribution in the arts. In 2010, I conducted the recreational jazz big band at the BISYOC Inter-Cultural Youth Orchestra Exchange in Ludlow, UK.

Mentoring Experience: Mentoring students has been one of the most fulfilling aspects of my Ph.D. Through the MIT SuperUROP program I mentored two year-long undergraduate research projects. I guided each student in finding their own research questions, designing experiments to provide insights, developing strategies to deal with roadblocks, and communicating their findings. I continue to mentor and collaborate with new graduate students who join the broader Swarm project, the research that I lead. Through discussion and deliberation, I help them design, implement, and evaluate new techniques. Across all of these experiences, I observed that students are happiest and perform best when they are excited about their project. This takes time and requires openness and patience. For example, one of the undergraduate students produced a promising memory-reference-tracking tool after one month of work, but he really flourished for the subsequent seven months when he moved to hacking on LLVM to automatically exploit locality among parallel tasks. Systems hacking is his passion, and we found the right match. These experiences have been highly rewarding, and I look forward to continuing to mentor promising students as a faculty member.
**Teaching Philosophy:** My goal is to enable students to develop the competence and confidence to become an independent and collaborative learner and problem solver. My approach is influenced by past teaching experience and by observing talented educators as both a student and assistant, across the fields of computer science, engineering, and music. It consists of three main facets.

*Build upon fundamentals.* I believe in growing a mental toolbox of fundamentals. As students learn to model and navigate complex systems, their task can be simplified by leaning on abstraction. My role is to present the students with fundamentals, and to help spot these principles in real-world systems. For example, my favorite element of MIT’s 6.823 is the frequent identification and solution of dependences on hardware structures or register/memory data. Computer architects typically solve these dependences with one of three solutions: stalling, bypassing, or speculation. As we encounter a problem in a microarchitecture design, we pause, point out that the action of one module depends on another, and ask students how this could be solved. By building on fundamentals, I help students gain confidence, helping them move from what is known into what is unknown. With time they really thrive, as they grow their toolbox both independently and through collaboration with fellow students.

*Encourage critical thinking.* In problems spanning mathematics to computer systems, there are often several valid solutions. Building on their fundamentals, I encourage students to identify a range of alternate solutions, and to consider and evaluate their trade-offs. For an example in mathematics, a number of approaches are valid to prove a given statement or solve a given problem. What insights or variable substitutions made one approach more terse than the others? Did another approach build upon fewer fundamentals (e.g., chain and product rule) requiring less memorization? For an example in computer systems, modern processors issue loads and stores to virtual addresses, but these must be translated into physical addresses. What are the different concerns among *(i)* performing translation before accessing the L1 cache, *(ii)* delaying translation after accessing the L1, or *(iii)* something in between? Critical analysis invokes a deeper understanding of a topic by the student, and is a skill and practice that is crucial to any science or engineering career.

*Inspire and engage.* Among even the strongest-willed, remaining focused is difficult and learning new material is a challenge. The onus to overcome these problems rests on both the student, who strives to remove distractions, but crucially on the teacher, who must inspire interest and be engaging. My most influential teachers demonstrated infectious passion for their topics, inspiring me to pursue the challenges of parallel computing for my research and to continue amateur drumming well into my Ph.D.

I engage my students through concise and clear communication, active learning, and relating topics to real-world practices. First, through course material, students are navigating complex systems for the first time. To avoid additional complexity, it is imperative to use my time and materials with them, in the classroom or one-on-one, effectively and efficiently. I strive to explain and illustrate concepts concisely and clearly. Second, I was introduced to active learning in MAT190 and ESC103. I would pose a math problem to the class, ask them to briefly think about a solution, subsequently pair with a partner to devise a solution, and finally share the solution with the class through discussion. I was also unintentionally practicing active learning through music instruction and conducting; music instilled the importance of two-way pedagogy. Positive constructive feedback is vital to this process. In lectures, I will engage students by periodically pausing to pose a relevant question to elicit discussion, as well as encouraging student presentations in senior classes. Third, students often crave real-world relevance of the materials they study. How this is delivered can depend on their academic stage and interests. For undergraduate students, I like to foreshadow how a topic will apply in industry or later courses. For example, caches are hardware structures that do not affect program correctness, but restructuring hot code to exploit locality can boost performance to improve user experience. For graduate students, I prefer to touch on recent research papers as well as foundational classics, and to structure courses to include a major research project. Through flexible group projects, students can collaborate to gain initial insights into cutting-edge research problems. This may motivate them to continue the research beyond the course and promote collaboration among research groups.